gen4-4DPi Series

gen4-4DPi-43T (4.3 " Resistive Touch)
gen4-4DPi-50T (5.0 " Resistive Touch)
gen4-4DPi-70T (7.0 " Resistive Touch)
gen4-4DPi-43CT-CLB (4.3 " Capacitive Touch w/ CLB)
gen4-4DPi-50CT-CLB (5.0 " Capacitive Touch w/ CLB)
gen4-4DPi-70CT-CLB (7.0 " Capacitive Touch w/ CLB)

Datasheet

Revision 1.8

Contents

1. Description 4
2. Features 5
3. Pin Configuration and Summary 6
4. Connecting the Display to the Pi 10
   4.1. Hardware Connection 10
   4.2. Software Download/Installation 11
   4.3. Calibrating the Touch Screen 13
   4.4. Change the Display Orientation 14
   4.5. gen4-4DPi-Adaptor Push Button 14
   4.6. SPI Frequency and Compression 15
   4.7. Backlight Control 15
   4.8. Parameters Listing 16
   4.9. HDMI or 4DPi Output 16
   4.10. DPI Adjustment 17
5. Display Module Part Numbers 18
6. Cover Lens Bezel Mounting 18
7. Standard FFC Cable Specification 18
8. Latest Kernel Versions 19
9. Mechanical Details gen4-4DPi-43T 20
10. Mechanical Details gen4-4DPi-50T 21
11. Mechanical Details gen4-4DPi-70T 22
12. Mechanical Details gen4-4DPi-43CT-CLB 23
13. Mechanical Details gen4-4DPi-50CT-CLB 24
14. Mechanical Details gen4-4DPi-70CT-CLB 25
15. Schematic Diagram gen4-4DPi (Display module) 26
16. Schematic Diagram gen4-4DPi-Adaptor (Display Adaptor) 27
17. Specifications 28

18. Appendix 1 - Code Examples - Push Buttons 32
18.1. Example for communicating to Push Buttons, for C language 32
18.2. Example for communicating to Push Buttons, for Python language 34
18.3. Example for Shutdown and Reset buttons, for C language 35
18.4. Example for Shutdown and Reset buttons, for Python language 36
19. Revision History 38

1. Description

The gen4-4DPi range is the Primary Display for the Raspberry Pi A+, B+, Pi2, Pi3, Pi3 B+, Pi4, Pi Zero, Pi Zero W and Pi Zero 2 W, which displays the primary output of the Raspberry Pi, like what is normally sent to the HDMI or Composite output. It features an integrated Resistive Touch panel or Capacitive Touch panel, enabling the gen4-4DPi to function with the Raspberry Pi without the need for a mouse.

Communication between the gen4-4DPi and the Raspberry Pi is interfaced with a high-speed 48 MHz SPI connection, which uses an onboard processor for direct command interpretation and SPI communication compression, and features a customised DMA-enabled kernel. This combination allows this display to output a high frame rate compared to other SPI display solutions, when displaying a typical image/video, and can achieve higher depending on if the image can be compressed.

The gen4-4DPi is designed to work with the Raspberry Pi Operating System (previously named Raspbian OS) running on the Raspberry Pi, as that is the official Raspberry Pi operating system. It is also compatible with Pixel and Scratch.

Note

The display resolution of the 4.3" is 480x272 pixels, while the 5.0" and the 7.0" are 800x480 pixels, and thus may not display all menus on the desktop fully, without some downscaling.

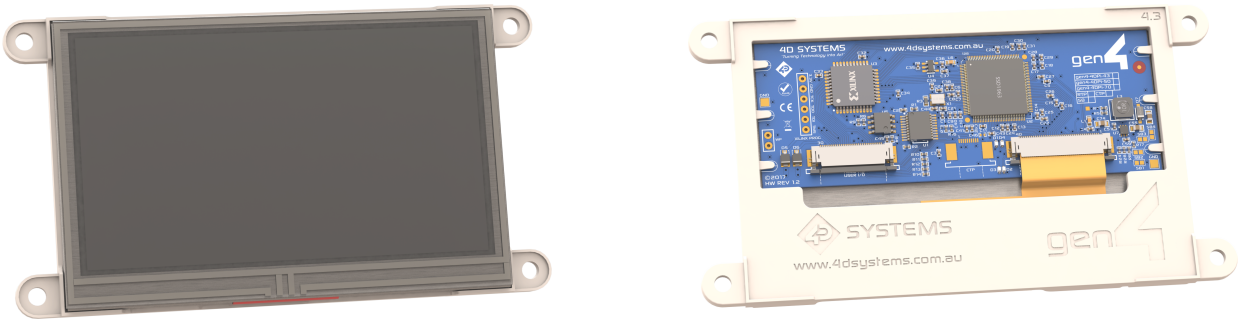
The gen4-4DPi range connects to the Raspberry Pi's 40-pin header using the gen4-4DPi Adaptor, which then connects to the gen4-4DPi display module using a 30-way FFC Cable.

Note

Raspberry Pi is a trademark of the Raspberry Pi Foundation, and all references to the words 'Raspberry Pi' or the use of its logo/marks are strictly about the Raspberry Pi product, and how *this* product is compatible with but is not associated with the Raspberry Pi Foundation in any way.

2. Features

- Universal Primary Display for the Raspberry Pi.
- Compatible with Raspberry Pi A+, B+, Pi2, Pi3, Pi3 B+, Pi4, Pi Zero, Pi Zero W and Pi Zero 2 W.
- 480x272 Resolution (4.3")
- 800x480 Resolution (5.0" & 7.0")
- TFT Screen with integrated 4-wire Resistive Touch Panel (T), or Capacitive Touch Panel (CT) with Cover Lens Bezel (CLB).
- Display GUI output / primary output, just like a monitor connected to the Raspberry Pi
- High Speed 48MHz SPI connection to the Raspberry Pi, featuring SPI compression technology.
- The typical frame rate of 20 Frames per second (FPS) - 4.3", or 7 Frames per second (5" & 7"), higher if the image can be compressed further by the kernel. Lower if no compression is possible.
- Powered directly off the Raspberry Pi, no external power supply is required.
- On board EEPROM for board identification, following the HAT standard.
- 4 x 4.0mm Mounting holes on Non-Touch and Resistive Touch modules, or via adhesive for the Capacitive Touch model.
- RoHS and CE Compliant

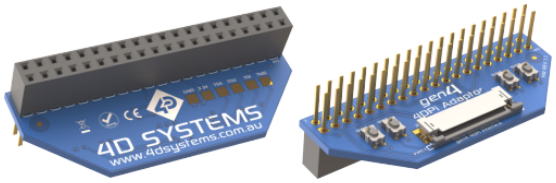


Resistive Touch Display configuration shown.

Note

CE EMC has not been conducted on these modules.

3. Pin Configuration and Summary



H1 Pinout (Raspberry Pi Connector on gen4-4DPi-Adaptor) – Female Connector

Pin	Symbol	I/O	Description
1	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi.
2	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi.
3	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi.
4	SDA1	I/O	I2C SDA1
5	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi.
6	SCL1	O	I2C SCL1
7	GPIO14	I/O	GPIO on the Raspberry Pi - **unused**
8	GPIO4	I/O	GPIO on the Raspberry Pi - **unused**
9	GPIO15	I/O	GPIO on the Raspberry Pi - **unused**
10	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi.
11	GPIO18	I/O	GPIO on the Raspberry Pi - Can be used for PWM Backlight, else **unused**
12	PENIRQ	I	Interrupt for the touchscreen controller
13	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
14	KEYIRQ	I	Interrupt for the push buttons
15	GPIO23	I/O	GPIO on the Raspberry Pi - **unused**
16	GPIO22	I/O	GPIO on the Raspberry Pi - **unused**
17	GPIO24	I/O	GPIO on the Raspberry Pi - **unused**
18	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi
19	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi

Pin	Symbol	I/O	Description
20	MOSI	O	SPI MOSI Pin
21	GPIO25	I/O	GPIO on the Raspberry Pi - **unused**
22	MISO	I/O	SPI MISO Pin
23	SPI-CS0	O	SPI Chip Select 0 – Used for Xilinx Processor for Display, to Raspberry Pi
24	SCK	O	SPI SCK Clock Pin
25	SPI-CS1	O	SPI Chip Select 1 – unused
26	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
27	ID-SC	O	I2C ID EEPROM
28	ID-SD	I/O	I2C ID EEPROM
29	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
30	GPIO	I/O	GPIO on the Raspberry Pi - **unused**
31	GPIO2	I/O	GPIO on the Raspberry Pi - **unused**
32	GPIO6	I/O	GPIO on the Raspberry Pi - unused
33	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
34	GPIO13	I/O	GPIO on the Raspberry Pi - **unused**
35	GPIO16	I/O	GPIO on the Raspberry Pi - **unused**
36	GPIO19	I/O	GPIO on the Raspberry Pi - **unused**
37	GPIO20	I/O	GPIO on the Raspberry Pi - **unused**
38	GPIO26	I/O	GPIO on the Raspberry Pi - **unused**
39	GPIO21	I/O	GPIO on the Raspberry Pi - **unused**
40	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi

Note

- **I** = Input, **O** = Output, **P** = Power
- The onboard Xilinx processor of the gen4-4DPi uses one of the Chip Select (CS) pins on the Raspberry Pi's SPI Bus (SPI-CS0). There is SPI-CS1 still available for use by the User.
- The onboard Resistive Touch Screen Controller or on-board Capacitive Touch Controller, uses the I2C bus (SDA1, SCL1) to communicate with the Raspberry Pi. The I2C bus is capable of communicating with other devices also, so isn't restricted to only the 4DPi's touch controller.

gen4-4DPi 30-way FFC Interface, between gen4-4DPi-Adaptor and gen4-4DPi Display

Pin	Symbol	I/O	Description
1	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
2	SDA1	I/O	I2C SDA1
3	SCL1	O	I2C SCL1
4	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
5	MOSI	O	SPI MOSI Pin
6	SCK	O	SPI SCK Clock Pin
7	MISO	I	SPI MISO Pin
8	SPI-CS0	O	SPI Chip Select 0 – Used for Xilinx Processor for Display, to Raspberry Pi
9	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
10	ID-SC	O	I2C ID EEPROM
11	ID-SD	I/O	I2C ID EEPROM
12	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
13	PENTRQ	I	Interrupt for the touchscreen controller
14	KEYIRQ	I	Interrupt for the push buttons
15	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
16	SW5	I	Button 5 (Not present on Adaptor), connected to Xilinx Processor on display
17	SW4	I	Button 4 on gen4-4DPi-Adaptor, connected to Xilinx Processor on display
18	SW3	I	Button 3 on gen4-4DPi-Adaptor, connected to Xilinx Processor on display
19	SW2	I	Button 2 on gen4-4DPi-Adaptor, connected to Xilinx Processor on display
20	SW1	I	Button 1 on gen4-4DPi-Adaptor, connected to Xilinx Processor on display
21	JTAG-TMS	--	Special Pins for Factory Programming of Xilinx Processor only – **Do Not Connect**
22	JTAG-TDI	--	Special Pins for Factory Programming of Xilinx Processor only – **Do Not Connect**

Pin	Symbol	I/O	Description
23	JTAG-TDO	--	Special Pins for Factory Programming of Xilinx Processor only – **Do Not Connect**
24	JTAG-TCK	--	Special Pins for Factory Programming of Xilinx Processor only – **Do Not Connect**
25	GND	P	Ground Pin, connected to the main system Ground of the Raspberry Pi
26	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi
27	+5V	P	+5V Supply Pin, connected to the main 5V supply of the Raspberry Pi
28	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi
29	+3.3V	P	+3.3V Supply Pin, connected to the main 3.3V supply of the Raspberry Pi
30	JTAG-TDO	GND	Ground Pin, connected to the main system Ground of the Raspberry Pi

Note

I = Input, **O** = Output, **P** = Power

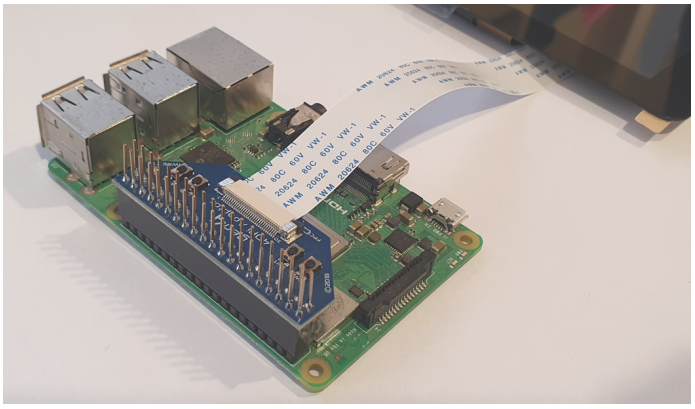
4. Connecting the Display to the Pi

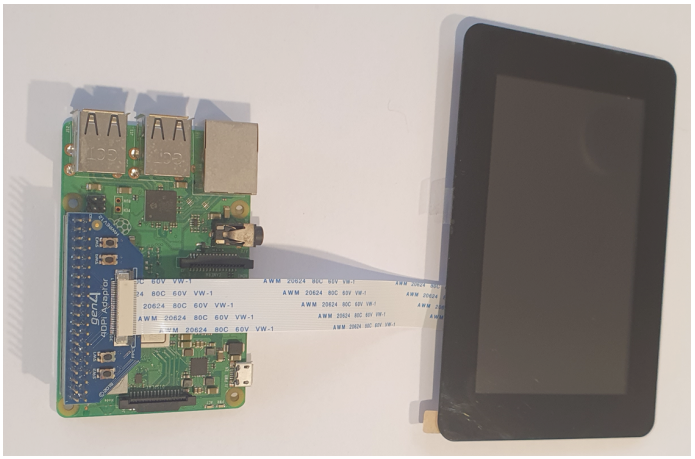
4.1. Hardware Connection

The gen4-4DPi is easily connected to a Raspberry Pi. Ensure the Raspberry Pi is powered off when connecting the gen4-4DPi display or adaptor.

Simply align the Female 40-way header on the gen4-4DPi-Adaptor with the Raspberry PI's Male 40-way header and connect them - ensuring the alignment is correct and all pins are seated fully and correctly. The gen4-4DPi-Adaptor should be overhanging inward of the Raspberry Pi.

Next simply connect the 30-way FFC cable between the FFC Connector of the gen4-4DPi and the gen4-4DPi-Adaptor, ensuring the copper pins of the FFC are facing upward in the connector.





Note

The hardware connected to the Pi is not recommended until the Pi has been set up. Please see the instructions below.

4.2. Software Download/Installation

4D Systems has prepared a custom DMA-enabled kernel for use with the Raspberry Pi Operating System (previously named Raspbian OS), which is available for download as a single package. This can be installed over your existing OS installation, or it can be applied over a fresh image. We **recommend** that you apply it over a fresh image.

If you are starting from a fresh image, start from Step 1, or else skip to step 3 if you already have an OS image and want to apply this kernel to that. If you are not installing from a fresh image and you encounter issues, we won't know the settings of your OS so please try and use a fresh image to determine any modifications that conflict with our kernel release. If you are running an OS with a Kernel version later than our Kernel Pack, you might encounter problems. Please contact support if you have problems. If you already have a custom Kernel, then applying our Kernel Pack over your custom Kernel will likely stop your previous modifications from working. You will need to build the kernel from scratch using the steps below.

STEPS (recommendations):

1. Install a fresh operating system as discussed on the Raspberry Pi website. Enable SSH and Wi-Fi as preferred.
2. Connect the gen4-4DPi and insert the uSD card into the Raspberry Pi. You will need network connectivity to proceed with the installation. A monitor, keyboard and mouse are required if not using SSH. SSH can be configured in Step 1. Power on the Raspberry Pi and make sure it is connected to your network.
3. Login to the Raspberry Pi using the standard 'pi' and 'raspberry' credentials or as configured in Step 1. If SSH is not used, open the Terminal app.
4. You are welcome to perform a system update if prompted, but please take note that **if you install a newer kernel than what our Kernel Pack offers, then you could encounter problems.** Therefore, it's not advisable to update the system, as this could update the Kernel.
5. Typically, on modern versions of the Pi OS, this following step is not required or is done automatically. However, it is here for reference. Expand the file system on the downloaded image using raspi-config (submenu Expand Filesystem). After exiting raspi-config a reboot is needed.

   ```
   $ sudo raspi-config
   $ sudo reboot
   ```

6. Once rebooted, you need to do an `apt-get upgrade`, because doing this after applying the Kernel Pack will render the 4DPi modifications disabled. Please note that doing an upgrade could change your current Kernel which could make the version installed newer than the Kernel Pack you are about to install next. The Kernel pack must be applied to a kernel very close (a newer Kernel Pack is generally OK) if not identical to the kernel your OS is running, or there will be issues.
7. Log into your Raspberry Pi again, you will need to download and install the Kernel Pack which supports the gen4-4DPi. The following step requires `sudo root` access.

8. To download and install files, enter the following commands in terminal/shell/SSH to download the kernel from the 4D Systems Server:

```
$ wget https://4dsystems.com.au/download/14644/ -O gen4-hats.tar.gz
```

   - Then extract the kernel pack:

```
$ sudo tar --keep-directory-symlink -xzvf gen4-hats.tar.gz -C /
```

   - If you encounter issues running the above command, try adding `--no-same-owner`

```
$ sudo tar --no-same-owner --keep-directory-symlink -xzvf gen4-hats.tar.gz -C /
```

   - The package selects the kernel required for the Raspberry Pi model, automatically. If you want to check for the kernel packages released by 4D systems, proceed to the Latest Kernel Versions section.

9. Reboot the Raspberry Pi by running the command

```
sudo reboot now
```

10. The desktop should begin to show on the gen4-4DPi once the Raspberry Pi has booted.
11. Doing an `apt-get upgrade` after the Kernel Pack has been installed, will disable the 4DPi and its modifications, as the modules and Kernel would be updated in this process. To reenable, be sure to download the latest Kernel Pack (check this datasheet again if there has been an updated version) and perform the same steps to get up and running again. Results may vary, and it's always advisable to apply the 4DPi Kernel Pack to a fresh image, but this is not always possible.

> **Warning**
>
> An upgrade should only be done if the latest RPi OS kernel is supported by the latest 4D kernel pack. Otherwise, installing the 4D kernel pack will downgrade the kernel and problems may occur.

12. **ADVANCED USERS:** If you need to make custom modifications to your Kernel, and want the 4DPi to function, you will need to build the Kernel from the source, and include the 4DPi files in the process. The link to our source is on our website, along with the steps required to add in the 4DPi files so this can be enabled in *menuconfig* while building the Kernel.

> **Note**
>
> - It is advisable to use the RPi OS release with a matching kernel version as one of the latest 4DPi packages that you plan to use. If support for a newer version is not yet available, please raise a ticket for assistance.
> - RPi OS based on Debian Bullseye is not fully compatible with our kernel release. Please use the latest Legacy version which is based on Debian Buster instead.

4.3. Calibrating the Touch Screen

Each gen4-4DPi which is shipped from the 4D Systems factory is slightly different, in the sense that each of the touch screens has a slightly different calibration. To get the best from your gen4-4DPi, you will need to calibrate the display, so it is as accurate as possible. This is typically only required for Resistive Touch models.

To calibrate the touch screen, the xinput_calibrator is required, and the following steps should be carried out. Make sure the Desktop is not running before you start, quit the desktop if it is and return to the terminal prompt.

> **Note**
>
> Only resistive touch display modules could be calibrated.

1. Install xinput_calibrator (if not installed by default) by running this command in the terminal.

```
sudo apt-get install xinput-calibrator
```

2. Install the event device input driver:

```
sudo apt-get install xserver-xorg-input-evdev
```

3. Rename *10-evdev.conf* file to *45-evdev.conf*.

```
sudo mv /usr/share/X11/xorg.conf.d/10-evdev.conf /usr/share/X11/xorg.conf.d/45-evdev.conf
```

4. Check if *evdev.conf* has a higher number than *libinput.conf*.

```
ls /usr/share/X11/xorg.conf.d/
```

   - The user should get something like this:

```
10-quirks.conf 40-libinput.conf 45-evdev.conf 99-fbturbo.conf
```

5. Perform a reboot

```
sudo reboot now
```

6. Reconnect to SSH and run xinput calibrator.

```
DISPLAY=:0.0 xinput_calibrator
```

   - Perform the calibration and copy results. The result should be something like this:

```
Section "InputClass"
    Identifier "calibration"
    MatchProduct "AR1020 Touchscreen"
    Option "Calibration" "98 4001 175 3840"
    Option "SwapAxes" "0"
EndSection
```

7. You may test the changes after xinput calibrator ends. To make the changes permanent, paste the results to `/etc/X11/xorg.conf.d/99-calibration.conf`.

```
sudo nano /etc/X11/xorg.conf.d/99-calibration.conf
```

8. Save the file and perform a reboot

```
sudo reboot now
```

9. The Display should now be calibrated.

4.4. Change the Display Orientation

To change the display orientation, simply edit the */boot/cmdline.txt* file

Add the parameter below after the console parts in the parameter list:

```
4d_hats.rotate = 90
```

And change this to have the value of 0, 90, 180 or 270. It should look something like this:

```
console=serial0,115200 console=tty1 4d_hats.rotate=90 root= (etc etc)
```

Save the file and restart your Raspberry Pi.

The touch screen will automatically remap the alignment thanks to the custom kernel.

After changing the Display Orientation, you need to calibrate again the screen.

4.5. gen4-4DPi-Adaptor Push Button

The gen4-4DPi-Adaptor features 4 push buttons, which are connected to the Xilinx Processor. These can be used to trigger events on the Raspberry Pi. Please refer to the Appendix for code examples on how to utilize these buttons.

4.6. SPI Frequency and Compression

The gen4-4DPi can be adjusted to work with a range of SPI Frequencies and levels of compression, depending on the requirements of the product/project.

Increasing the frequency can result in a higher Frame Rate (FPS), however, will use more power and processor time. Increasing the level of compression can also result in a higher FPS but may cause the display to corrupt. By default, an SPI Frequency of 48Mhz is used, with a Compression level of 7.

The following parameters are the defaults in the */boot/cmdline.txt* file and can be edited to adjust the Frequency and Compression level.

```
4d_hats.sclk=48000000
4d_hats.compress=7
```

Setting compress to 1 will enable the kernel to control the level of compression based on the frequency selected. This however is not guaranteed to have a good result and may require manually setting the compression level of corruption on the display is experienced.

If corruption or display anomalies occur at any given compression level, try to lower it by 1 value and check if this has improved.

> **Note**
>
> Changing the frequency and compression requires a restart of the Raspberry Pi.

4.7. Backlight Control

The backlight brightness can be controlled from the terminal or a bash script. The following command can be used to set the backlight from 0 to 100%.

```
sudo sh -c 'echo 31 > /sys/class/backlight/4d-hats/brightness'
```

The above will set the backlight to 100% (default). Simply change the 'echo 31' to anything from 0 to 31.

4.8. Parameters Listing

The following is a list of all the custom parameters used by the gen4-4DPi.

rotate: screen rotation 0/90/180/270 (int)

compress: SPI compression 0/1/2/3/4/5/6/7 (int)

sclk: SPI clock frequency (long)

Valid SPI Frequency values (4d-hats.sclk):

Values can be almost anything. This has been tested up to 64Mhz. Common values would include 64000000 (64MHz), 48000000 (Default), 32000000, 24000000 etc.

Valid Compression values (4d-hats.compress):

0 (compression off)

1 (compression on, auto set based on sclk value)

2 (lowest), 3, 4, 5, 6, 7 (highest compression)

These parameters can be set or read from the */boot/cmdline.txt* file, and they can be read from the */sys/module/4d_hats/parameters/* directory.

For example:

```
cat /sys/module/4d_hats/parameters/rotate
```

Will display the current rotation saved.

4.9. HDMI or 4DPi Output

To switch the X Windows output being displayed on 4DPi or HDMI output, X can be launched using either of the following commands:

```
startx -- -layout TFT
startx -- -layout HDMI
```

Alternatively, these commands do the same thing:

```
FRAMEBUFFER=/dev/fb1 startx
startx
```

4.10. DPI Adjustment

It is possible to change the DPI output of the 4DPi in the same way as other LXDE-based systems.

- Login as pi and open terminal
- Check the current DPI settings by running this command:

```
xrdb -query -all
```

- The current dpi is listed next to the **Xft.dpi** listing.
- You can change the DPI by doing this.
  - Edit the following file, and then merge it:

    ```
    nano ~/.Xresources
    ```

  - Add this line to the file: `Xft.dpi: 75`.
  - This will set the DPI to **75**.
  - Save and exit the file.
  - Merge it so the value gets used, by doing the following:

    ```
    xrdb -merge ~/.Xresources
    ```

- You can now check the DPI settings.

```
xrdb -query -all
```

- Reboot the Pi, and your changes should take effect.

Changing the DPI can make the screen blurry, so take care when adjusting these values. If you get to a point where it is unreadable, SSH into your Pi and change the value back to something reasonable.

Ideally, DPI is set based on your resolution, however, for small-resolution displays, it can be desirable to make the DPI smaller so you can fit more on the screen.

5. Display Module Part Numbers

The following is a breakdown of the part numbers and what they mean.

Example: gen4-4DPi-70CT-CLB

- **gen4** - gen4 Display Range
- **4DPi** - Display Family
- **70** - Display size (7.0")
- **T** - Resistive Touch.
- **CT** - Capacitive Touch
- **CLB** - Cover Lens Bezel

> **Note**
>
> - For part numbers that do not include T or CT, these are non-touch variants.
> - Cover Lens Bezels (CLB) are glass fronts for the display module with overhanging edges, which allow the display module to be mounted directly into a panel using special adhesive on the overhanging glass. These are available for Capacitive Touch only.

6. Cover Lens Bezel Mounting

The perimeter of the CLB display modules features double-sided adhesive tape, designed to stick directly onto a panel, enclosure, box etc without the need for any mounting screws or hardware.

The tape used is 3M 9495LE tape, which uses well-known and strong 3M 300LSE adhesives. The double-sided adhesive has a thickness of 0.17mm once the backing has been removed. More information on this adhesive can be found on the 3M website.

7. Standard FFC Cable Specification

Between the gen4-4DPi-Adaptor and the gen4-4DPi Display Module, the following FFC cable is supplied:

- 30 Pin Flexible Flat Cable, 150mm Long, 0.5mm (0.02") pitch
- Cable Type: AWM 20624 80C 60V VW-1
- Heat Resistance 80 Degrees Celsius
- Connections on the **opposite side** at each end (Type B)

All FFC connectors on the gen4-4DPi are 'Top Contact' meaning the FFC cable has the metal 'fingers' pointing upward in the connector, blue stiffener on the back of the FFC cable is down on the PCB side.

8. Latest Kernel Versions

Here is the list of the kernel patches released by 4D systems.

Latest releases:

- gen4-hats_5-10-103.tar.gz
- gen4-hats_5-15-32_32bit.tar.gz (refer to the 2nd point from the note below)

Previous releases:

- gen4-hats_5-10-76-4DPi.tar.gz
- gen4-hats_5-10-63.tar.gz
- gen4-hats_5-4-68.tar.gz
- gen4-hats_4-19-57-v7l+_v1.0.tar.gz
- gen4-hats_4-14-34_v1.1.tar.gz
- gen4-hats_4-9-80_v1.1.tar.gz
- gen4-hats_4-9-59_v1.2.tar.gz

Note

1. It is highly advisable to use a Raspberry Pi OS release with **matching kernel version** (first 2 numbers and 3rd number need to be less than or equal) as the Kernel Pack you decide to use. Please refer to *step 1* under the Software Download/Installation section regarding current recommendations.

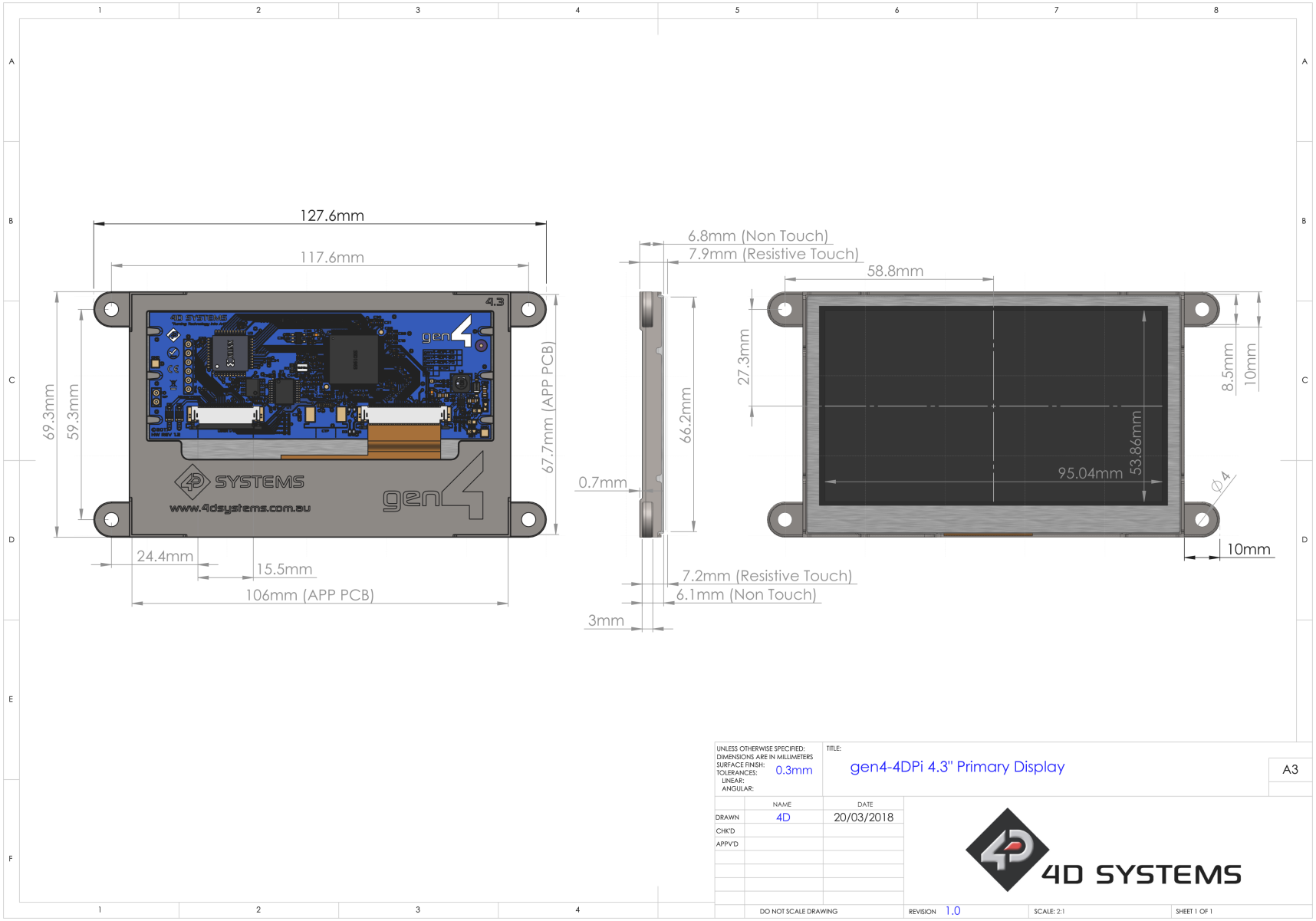
   For example, if your OS uses Kernel 5.4.60, then applying our 5.4.68 Kernel Pack is a good match.

   Example 2, If your OS uses 5.4.79, or 5.5.10, then applying the 5.4.68 Kernel Pack likely would not be the best idea as it would be a downgrade, and some things may not function correctly.
2. This kernel package is versions higher than the recommended latest Buster (legacy) version release. This was built to be able to update the Buster version to a matching kernel version of the Bullseye release. To match the kernel, you can use the command: `sudo rpi-update`

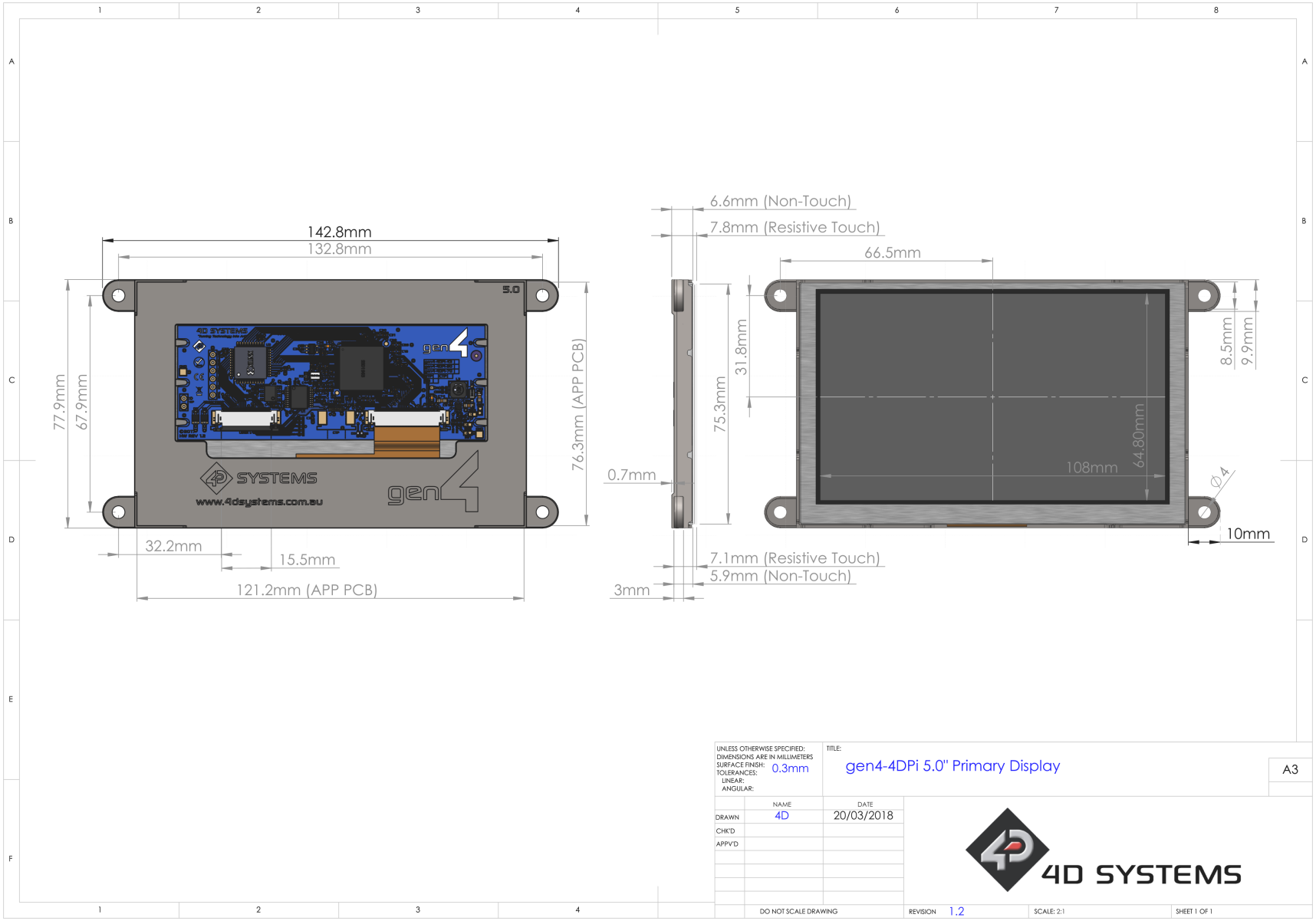
   ```
   sudo rpi-update <git hash>
   ```

   Git hash must be the commit in the Raspberry Pi Firmware Files repository with the same kernel version as the 4DPi package.
3. Some older kernel releases may be available upon request. Please contact 4D Systems Support Team for more information.

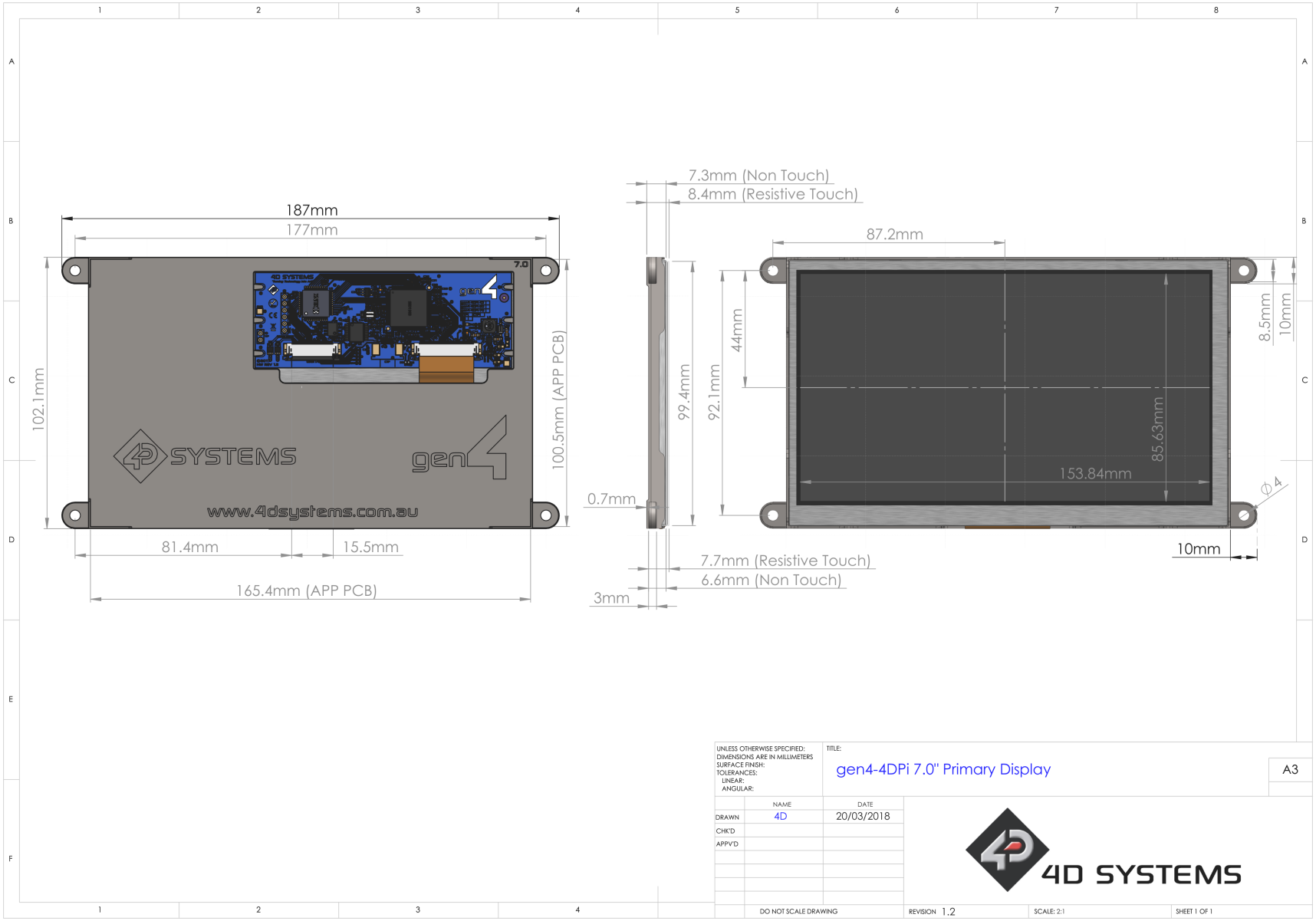
9. Mechanical Details gen4-4DPi-43T



10. Mechanical Details gen4-4DPi-50T



11. Mechanical Details gen4-4DPi-70T

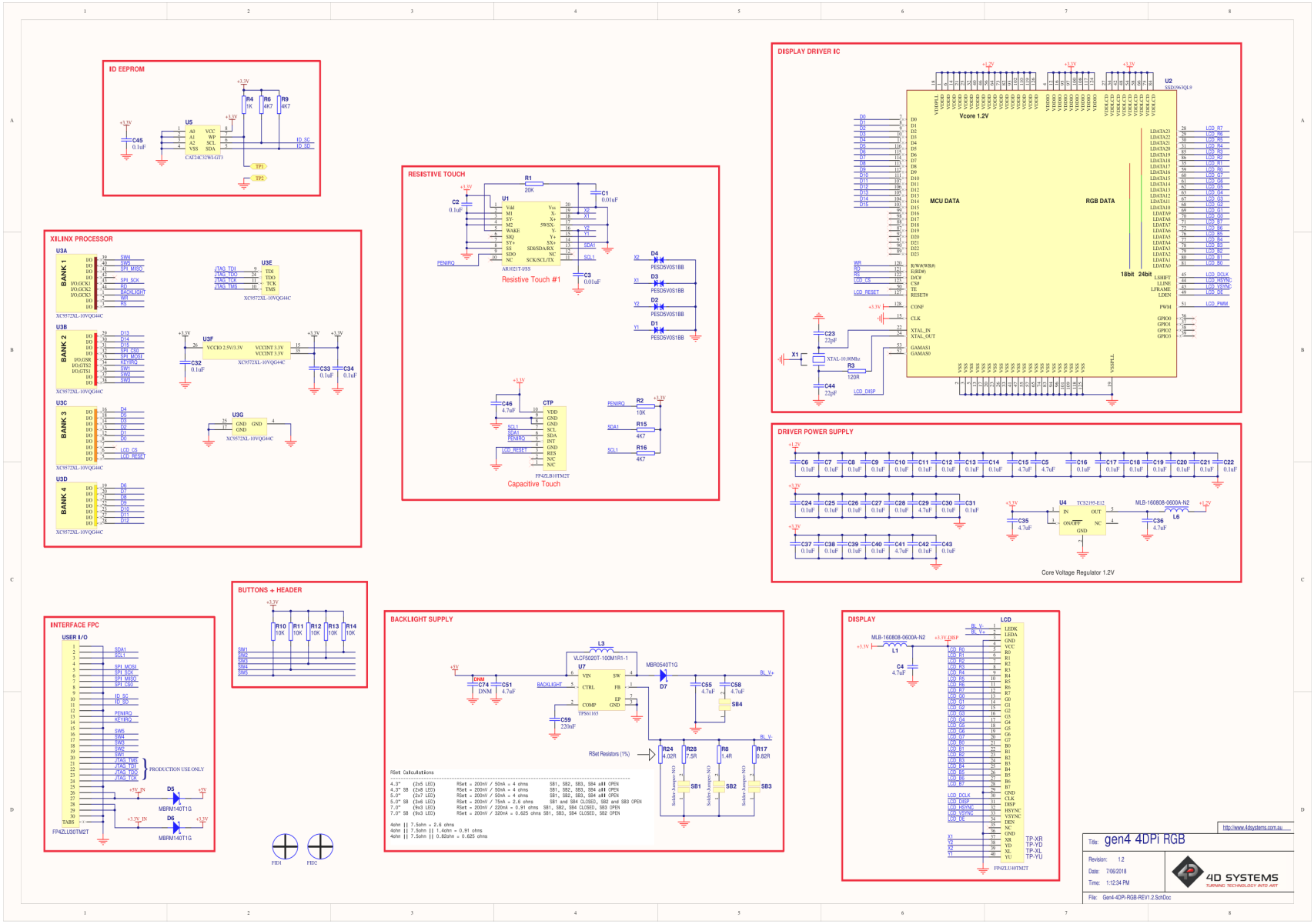


12. Mechanical Details gen4-4DPi-43CT-CLB

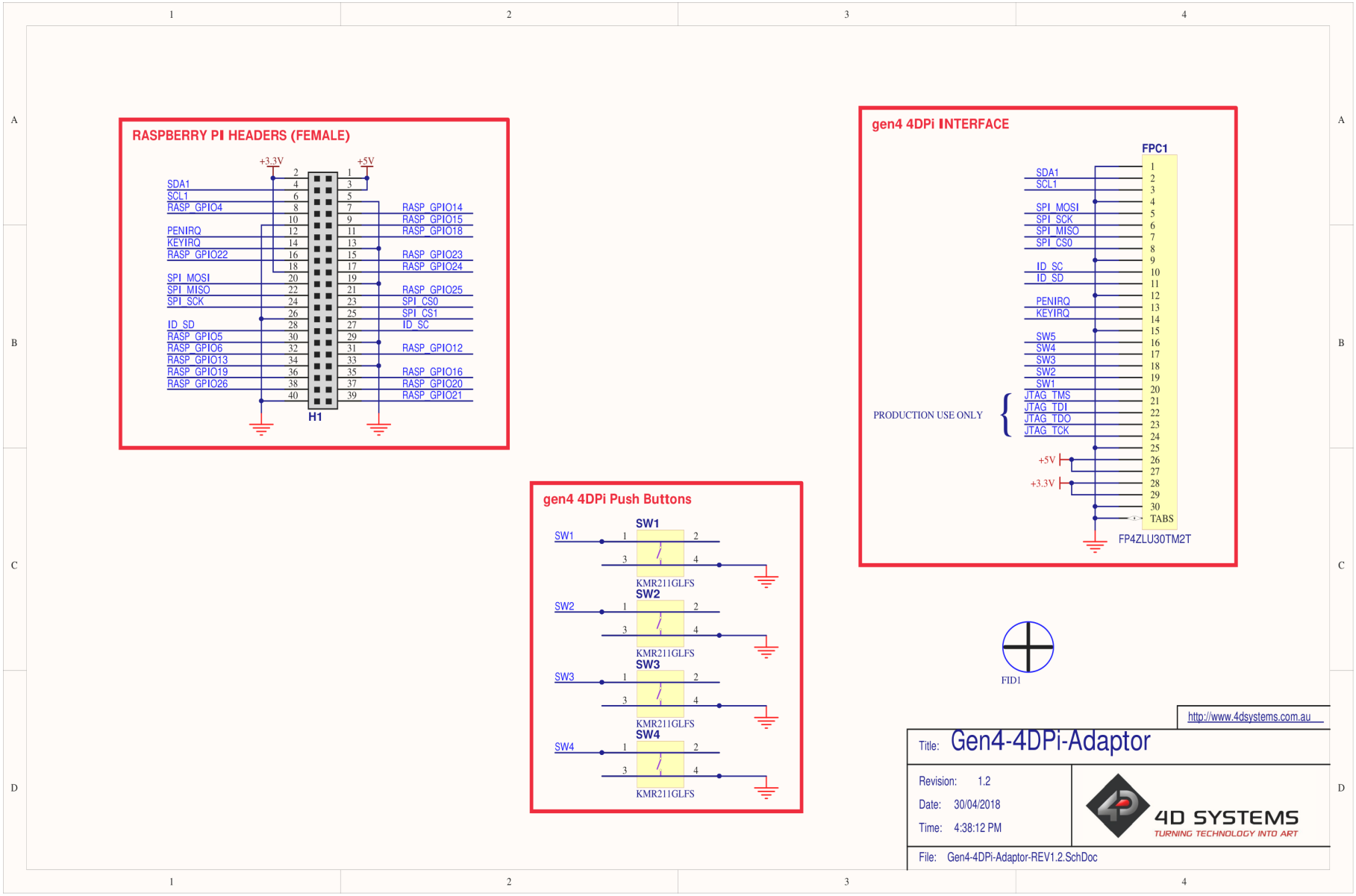
13. Mechanical Details gen4-4DPi-50CT-CLB

14. Mechanical Details gen4-4DPi-70CT-CLB

15. Schematic Diagram gen4-4DPi (Display module)



16. Schematic Diagram gen4-4DPi-Adaptor (Display Adaptor)



17. Specifications

Absolute Maximum Ratings

Operating ambient temperature	-20°C to +70°C
Storage temperature	-30°C to +80°C

Note

Stresses above those listed here may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the recommended operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Recommended Operating Conditions

Parameter	Conditions	Min	Typ	Max	Units
Supply Voltage (+3.3V)	Stable supply from Raspberry Pi Bus	3.0	3.3	4.0	V
Supply Voltage (+5V)	Stable supply from Raspberry Pi Bus	4.5	5.0	5.5	V

Global Characteristics Based on Operating Conditions

Parameter	Conditions	Min	Typ	Max	Units
Supply Current Estimates (Display portion only) (RPi current not included) Idle home screen	gen4-4DPi-43T (Max Brightness)	--	250	--	mA
	gen4-4DPi-43CT-CLB (Max Brightness)	--	270	--	mA
	gen4-4DPi-50T (Max Brightness)	--	360	--	mA
	gen4-4DPi-50CT-CLB (Max Brightness)	--	380	--	mA
	gen4-4DPi-70T (Max Brightness)	--	640	--	mA
		--	750	--	mA

Parameter	Conditions	Min	Typ	Max	Units
	gen4-4DPi-70CT-CLB (Max Brightness)				
Display Endurance	Hours of operation, measured to when the display is 50% original brightness	30000	--	--	H
Touch Screen Endurance (Resistive Touch)	Number of touches/hits with a 12.5mm tip at a rate of 2x per second with 250gf force	--	1M	--	Touches
	Slide stylus on screen, 100gf force, 60mm/s speed with a 0.8mm polyacetal tip stylus pen	--	100K	--	Slides
Touch Screen Transparency	Resistive Touch	82	--	--	%
	Capacitive Touch	90	--	--	%
Touch Screen Operational Force (Resistive Touch)	Only use Finger or Stylus, do not use anything sharp or metal	20	--	100	Gf
CLB Hardness (Capacitive Touch)	Cover Lens Bezel Glass Hardness	--	6	--	H

LCD Display Information (TN Display)

Parameter	Conditions	Specification
Display Type	All displays produced before the IPS release date in the IPS table below, or any modules not specified are therefore TN type displays.	**TN** - TFT Transmissive LCD
Display Sizes		4.3", 5.0" or 7.0" Diagonal
Display Resolution		480 x 272 (Landscape Viewing) – 4.3" 800 x 480 (Landscape Viewing) – 5.0" & 7.0"
Display Brightness	gen4-4DPi-43T (Max Brightness)	400 cd/m2
	gen4-4DPi-43CT-CLB (Max Brightness)	475 cd/m2

Parameter	Conditions	Specification
	gen4-4DPi-50T (Max Brightness)	400 cd/m2
	gen4-4DPi-50CT-CLB (Max Brightness)	475 cd/m2
	gen4-4DPi-70T (Max Brightness)	400 cd/m2
	gen4-4DPi-70CT-CLB (Max Brightness)	475 cd/m2
Display Contrast Ratio	Typical	500:1
Display Viewing Angles	Above Centre	70 Degrees
	Below Centre	60 Degrees
	Left of Centre	70 Degrees
	Right of Centre	70 Degrees
Display Viewing Direction	TN displays	12 O'clock Display (Optimal viewing is from above when in Landscape/Wide mode)
Display Backlighting	gen4-4DPi-43xx Models	2x5 Parallel LED's
	gen4-4DPi-50xx Models	2x6 Parallel LED's
	gen4-4DPi-70xx Models	9x3 Parallel LED's
Pixel Pitch	4.3"	0.198 x 0.198mm (Square pixels)
	5.0"	0.135 x 0.135mm (Square pixels)
	7.0"	0.1925 x 0.179mm (non-Square pixels)
Pixel Density (Number of pixels in 1 row in 25.4mm)	4.3"	128 DPI/PPI
	5.0"	183 DPI/PPI
	7.0"	132 DPI/PPI (Horizontal) 142 DPI/PPI (Vertical)

LCD Display Information (IPS Display)

Parameter	Conditions	Specification
Display Type	**gen4-4DPi-43T** – Modules produced from March 2021 onwards. **gen4-4DPi-43CT-CLB** – Modules produced from March 2021 onwards. **gen4-4DPi-50T** – Modules produced from March 2021 onwards. **gen4-4DPi-50CT-CLB** – Modules produced from September 2020 onwards.	**IPS** - TFT Transmissive LCD
Display Sizes		4.3", 5.0" or 7.0" Diagonal

Parameter	Conditions	Specification
Display Resolution		480 "x 272 (Landscape Viewing) – 4.3" 800 x 480 (Landscape Viewing) – 5.0" & 7.0"
Display Brightness	gen4-4DPi-43T (Max Brightness)	510 cd/m2
	gen4-4DPi-43CT-CLB (Max Brightness)	540 cd/m2
	gen4-4DPi-50T (Max Brightness)	510 cd/m2
	gen4-4DPi-50CT-CLB (Max Brightness)	540 cd/m2
Display Contrast Ratio	gen4-4DPi-43xx Models - Typical	800:1
	gen4-4DPi-50xx Models - Typical	1000:1
Display Viewing Angles	Above Centre	80 Degrees
	Below Centre	80 Degrees
	Left of Centre	80 Degrees
	Right of Centre	80 Degrees
Display Viewing Direction	IPS Displays	ALL (Viewing from all directions)
Display Backlighting	gen4-4DPi-43xx Models	2x5 Parallel LED's
	gen4-4DPi-50xx Models	3x6 Parallel LED's
Pixel Pitch	4.3"	0.198 x 0.198mm (Square pixels)
	5.0"	0.135 x 0.135mm (Square pixels)
Pixel Density (Number of pixels in 1 row in 25.4mm)	4.3"	128 DPI/PPI
	5.0"	183 DPI/PPI

Performance

Parameter	Conditions	Min	Typ	Max	Units
Frame Rate (FPS) (4.3" only)	Video Playback, Full Screen, 480x272. A higher FPS can be achieved if the display outputs lots of blocks of the same colour. See the SPI Frequency and Compression section	--	20	--	FPS

Parameter	Conditions	Min	Typ	Max	Units
Frame Rate (FPS) (5.0" & 7.0" only)	Video Playback, Full Screen, 800x480. A higher FPS can be achieved if the display outputs lots of blocks of the same colour. See the SPI Frequency and Compression section	--	7	--	FPS

Ordering Information

Order Code:
gen4-4DPi-43T
gen4-4DPi-50T
gen4-4DPi-70T
gen4-4DPi-43CT-CLB
gen4-4DPi-50CT-CLB
gen4-4DPi-70CT-CLB
Packaging: Module sealed in a 4D Systems Box

18. Appendix 1 - Code Examples - Push Buttons

18.1. Example for communicating to Push Buttons, for C language

```
// test program to read state of buttons on 4D Systems 4DPi displays

#include <stdio.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>

#define LCD4DPI_GET_KEYS _IOR('K', 1, unsigned char *)

void print_keys(int fd)
{
    unsigned char keys;

    if (ioctl(fd, LCD4DPI_GET_KEYS, &keys) == -1)
    {
        perror("_apps ioctl get");
```

```
    }
    else
    {
        printf("Keys : %2x\n", keys);
    }
}

int main(int argc, char *argv[])
{
    char *file_name = "/dev/fb1";
    int fd;

    fd = open(file_name, O_RDWR);
    if (fd == -1)
    {
        perror("_apps open");
        return 2;
    }

    print_keys(fd);
    printf("Ioctl Number: (dec)%d  (hex)%x\n", LCD4DPI_GET_KEYS, LCD4DPI_GET_KEYS);

    close (fd);
    return 0;
}
```

18.2. Example for communicating to Push Buttons, for Python language

```
#!/usr/bin/python

import array, fcntl
from time import sleep

# test program to read state of buttons on 4D Systems 4DPi displays

# LCD4DPI_GET_KEYS = -2147202303

_IOC_NRBITS = 8
_IOC_TYPEBITS = 8
_IOC_SIZEBITS = 14
_IOC_DIRBITS = 2
_IOC_DIRMASK = (1 << _IOC_DIRBITS) - 1
_IOC_NRMASK = (1 << _IOC_NRBITS) - 1
_IOC_TYPEMASK = (1 << _IOC_TYPEBITS ) - 1
_IOC_NRSHIFT = 0
_IOC_TYPESHIFT = _IOC_NRSHIFT+_IOC_NRBITS
_IOC_SIZESHIFT = _IOC_TYPESHIFT+_IOC_TYPEBITS
_IOC_DIRSHIFT = _IOC_SIZESHIFT+_IOC_SIZEBITS
_IOC_NONE = 0
_IOC_WRITE = 1
_IOC_READ = 2

def _IOC(dir, type, nr, size):
  # print 'dirshift {}, typeshift {}, nrshift {}, sizeshift {}'.format(_IOC_DIRSHIFT,
_IOC_TYPESHIFT, _IOC_NRSHIFT, _IOC_SIZESHIFT)
  ioc = (dir << _IOC_DIRSHIFT ) | (type << _IOC_TYPESHIFT ) | (nr << _IOC_NRSHIFT ) | (size <<
_IOC_SIZESHIFT)
  if ioc > 2147483647: ioc -= 4294967296
  return ioc

#def _IO(type, nr):
# return _IOC(_IOC_NONE, type, nr, 0)

def _IOR(type,nr,size):
  return _IOC(_IOC_READ, type, nr, size)

#def _IOW(type,nr,size):
# return _IOC(_IOC_WRITE, type, nr, sizeof(size))

LCD4DPI_GET_KEYS = _IOR(ord('K'), 1, 4)
buf = array.array('h',[0])

print 'Press Top & Bottom buttons simultaneously to exit'

with open('/dev/fb1', 'rw') as fd:

  while True:
    fcntl.ioctl(fd, LCD4DPI_GET_KEYS, buf, 1) # execute ioctl call to read the keys
    keys = buf[0]

    if not keys & 0b00001:
      print "KEY1" ,
    if not keys & 0b00010:
      print "KEY2" ,
    if not keys & 0b00100:
```

```
      print "KEY3" ,
    if not keys & 0b01000:
      print "KEY4" ,
    if not keys & 0b10000:
      print "KEY5" ,

    if keys != 0b11111:
      print
    if keys == 0b01110: # exit if top and bottom pressed
      break

    sleep(0.1)
```

18.3. Example for Shutdown and Reset buttons, for C language

```
// test program to Shutdown or Restart Pi using buttons on 4D Systems 4DPi displays

#include <stdio.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>

#define LCD4DPI_GET_KEYS _IOR('K', 1, unsigned char *)

int get_keys(int fd, unsigned char *keys)
{
    if (ioctl(fd, LCD4DPI_GET_KEYS, keys) == -1)
    {
        perror("_apps ioctl get");
    return 1;
    }
    *keys &= 0b11111;
    return 0;
}

int main(int argc, char *argv[])
{
    char *file_name = "/dev/fb1";
    int fd;
    unsigned char key_status;

    fd = open(file_name, O_RDWR);
    if (fd == -1)
    {
        perror("_apps open");
        return 2;
    }

    key_status = 0b11111;
    while(key_status & 0b00001) // press key 1 to exit
    {
        if(get_keys(fd, &key_status) != 0)
             break;

        // printf("key_status: %x\n", key_status);
```

```
        if(!(key_status & 0b10000))
        {
            system("sudo shutdown -h now");
            break;
        }

        if(!(key_status & 0b01000))
        {
            system("sudo reboot");
            break;
        }

        sleep(0.1);
    }

    close(fd);
    return 0;
}
```

18.4. Example for Shutdown and Reset buttons, for Python language

```
#!/usr/bin/python

import array, fcntl, os
from time import sleep
# test program to Shutdown or Restart Pi using buttons on 4D Systems 4DPi displays

#LCD4DPI_GET_KEYS = -2147202303

_IOC_NRBITS   =  8
_IOC_TYPEBITS =  8
_IOC_SIZEBITS = 14
_IOC_DIRBITS  =  2

_IOC_DIRMASK    = (1 << _IOC_DIRBITS) - 1
_IOC_NRMASK     = (1 << _IOC_NRBITS) - 1
_IOC_TYPEMASK   = (1 << _IOC_TYPEBITS ) - 1

_IOC_NRSHIFT = 0
_IOC_TYPESHIFT = _IOC_NRSHIFT+_IOC_NRBITS
_IOC_SIZESHIFT = _IOC_TYPESHIFT+_IOC_TYPEBITS
_IOC_DIRSHIFT  = _IOC_SIZESHIFT+_IOC_SIZEBITS

_IOC_NONE = 0
_IOC_WRITE = 1
_IOC_READ = 2

def _IOC(dir, type, nr, size):
  # print 'dirshift {}, typeshift {}, nrshift {}, sizeshift {}'.format(_IOC_DIRSHIFT,
_IOC_TYPESHIFT, _IOC_NRSHIFT, _IOC_SIZESHIFT)
  ioc = (dir << _IOC_DIRSHIFT ) | (type << _IOC_TYPESHIFT ) | (nr << _IOC_NRSHIFT ) | (size <<
_IOC_SIZESHIFT)
  if ioc > 2147483647: ioc -= 4294967296
  return ioc

#def _IO(type, nr):
#  return _IOC(_IOC_NONE,  type, nr, 0)
```

```
def _IOR(type,nr,size):
  return _IOC(_IOC_READ,  type, nr, size)

#def _IOW(type,nr,size):
#  return _IOC(_IOC_WRITE, type, nr, sizeof(size))

LCD4DPI_GET_KEYS = _IOR(ord('K'), 1, 4)
#print 'ssd {} {:12} {:0>8x} {:0>32b}'.format(ssd1289, hex(ssd1289), ssd1289, ssd1289)
buf = array.array('h',[0])

with open('/dev/fb1', 'rw') as fd:

  while True:
    fcntl.ioctl(fd, LCD4DPI_GET_KEYS, buf, 1) # execute ioctl call to read the keys
    keys = buf[0]

    if not keys & 0b00001:
      break
    if not keys & 0b10000:
      os.system("sudo shutdown -h now")
      break
    if not keys & 0b01000:
      os.system("sudo reboot")
      break

    sleep(0.1)
```